# 基于BP神经网络实现手写数字识别

## 实验目的

构建BP神经网络，实现手写数字的识别。

## 实验方法

### 搭建平台

使用Windows系统自带的画图工具分别构建训练样本集和测试样本集，使用MATLAB GUI编程环境对它们进行处理、识别。

### 特征描述

将手写部分从图像中分割，并转化为灰度图像，而后将其分割为6\*6=36个小图片，用每个小图片中黑色像素占总像素的比重作为该样本的特征，也就是一个36维的向量。

子函数get\_feature(image)代码如下：

function[feature] = get\_feature(image)

%从图像中分割出手写部分并转为灰度

image2 = pretreat(image);

A = mat2cell(image2,[10,10,10,10,10,10],[7,7,7,7,7,7]);

B = reshape(A,1,36);

for i=1:length(B)

temp = cell2mat(B(i));

%feature(i)表示黑像素点占一个小块的比例

feature(i) = sum(temp(:) ~= 255) / numel(temp);

end

%feature

end

### 关键算法

#### BP神经网络简介

BP(back propagation)神经网络是1986年由Rumelhart和McClelland为首的科学家提出的概念，是一种按照误差逆向传播算法训练的多层前馈神经网络，是目前应用最广泛的神经网络。

BP神经网络具有任意复杂的模式分类能力和优良的多维函数映射能力，解决了简单感知器不能解决的异或(Exclusive OR，XOR)和一些其他问题。从结构上讲，BP网络具有输入层、隐藏层和输出层；从本质上讲，BP算法就是以网络误差平方为目标函数、采用梯度下降法来计算目标函数的最小值。

基本BP算法包括信号的前向传播和误差的反向传播两个过程。即计算误差输出时按从输入到输出的方向进行，而调整权值和阈值则从输出到输入的方向进行。正向传播时，输入信号通过隐含层作用于输出节点，经过非线性变换，产生输出信号，若实际输出与期望输出不相符，则转入误差的反向传播过程。误差反传是将输出误差通过隐含层向输入层逐层反传，并将误差分摊给各层所有单元，以从各层获得的误差信号作为调整各单元权值的依据。通过调整输入节点与隐层节点的联接强度和隐层节点与输出节点的联接强度以及阈值，使误差沿梯度方向下降，经过反复学习训练，确定与最小误差相对应的网络参数(权值和阈值)，训练即告停止。

#### BP神经网络的训练方式

首先约定如下：假定神经网络由L层神经元组成，输入层有个节点，第r层有个神经元，其中。所有的神经元使用呢相同的Sigmoid函数，即。假定有N个可用的训练对，其中，。用表示第r层第j个神经元的权向量（包括阈值），用表示第r层第j个神经元的权值总和，用表示第r层第j个神经元激活函数的输出。

那么，BP神经网络的训练过程可表述如下：

##### 初始化：用伪随机序列产生器生成的小随机值初始化所有权值。

##### 1、前向计算：对于每个训练特征向量，用式计算所有的，其中。用式和估算当前的代价函数。

##### 2、后向计算：对于每个和，用式计算。对于每个和，用式和求出。

##### 修正权值。对于和，用式以及更新网络的权值。

#### 算法实现

上述过程用MATLAB实现如下：

%对网络进行训练

function [] = train()

%初始化，生成随机权值并保存样本数据

init();

%w1每一行都是隐层节点的权值，包括阈值

load w1

%w2每一行都是输出节点的权值，包括阈值

load w2

%feature\_map每一页都是一类样本的特征

load feature\_map

miu =0.9;%学习率

b = size(w1);

InNum = b(2)-1;%输入节点数

HiddenNum = b(1);%隐层节点数

b = size(w2);

OutNum = b(1);%输出节点数

%存储代价函数的值

eii=[];

%抽取1000个样本进行训练

for p=1:1000

load w1

load w2

delta\_w1=zeros(HiddenNum,InNum+1);

delta\_w2 = zeros(OutNum,HiddenNum+1);

%随机选取一类样本

i=randperm(10,1)-1;

%设置期望输出y，对于数字i的样本，希望输出中只有第i+1维的数字为1，其余为0

y=zeros(1,OutNum);

y(i+1)=1;

b = size(feature\_map(:,:,i+1));

%随机选取该类样本中的一个

j=randperm(b(1),1);

%获取样本特征

feature = feature\_map(j,:,i+1);

%下面进行前向计算

%先是隐层

y1=[];%隐层的输出

v1=[];%激活函数的输入

for m=1:HiddenNum

v1 = [v1,([1,feature])\*w1(m,:)'];

end

for m=1:HiddenNum

y1 = [y1,f(v1(m))];

end

%输出层

y2=[];%输出

v2=[];%输出层激活函数的输入

for n=1:OutNum

v2 = [v2,([1,y1])\*w2(n,:)'];

end

for n=1:OutNum

y2 = [y2,f(v2(n))];

end

%这里的y2就是神经网络的输出，前向计算完毕

%下面进行后向计算

%偏差

e = y2-y;

eii = [eii,0.5\*sum(e.^2)];

%对于输出层

delta2 = [];

for n=1:OutNum

delta2 = [delta2,e(n)\*f2(v2(n))];

end

%对于隐层

delta1 = [];

for m=1:HiddenNum

temp\_delta = delta2\*w2(:,m+1);

delta1 = [delta1,temp\_delta\*f2(v1(m))];

end

%该样本对修正过程的影响

for n=1:OutNum

delta\_w2(n,2:end) = delta\_w2(n,2:end) + delta2(n)\*y1;

end

for m=1:HiddenNum

delta\_w1(m,2:end) = delta\_w1(m,2:end) + delta1(m)\*feature;

end

%更新权值

w1 = w1+(-miu)\*delta\_w1;

w2 = w2+(-miu)\*delta\_w2;

save w1.mat w1

save w2.mat w2

end

plot(1:1000,eii);

end

## 实验结果

### 训练样本集构建

使用Windows系统自带的画图软件创建训练样本集，使用10个文件夹分别保存10个类的训练集，如下图。



图 三-1 训练样本集

每一个样本都有一个固定的模板图片，用画图工具在该图片的指定位置手写数字，模板图片如下图。

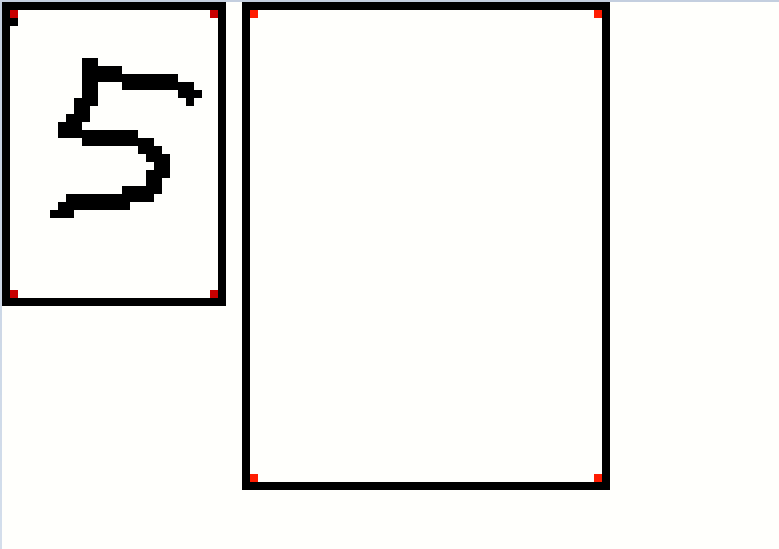


图 三-2 样本构建的模板图片

某一手写完成的样本如下图。

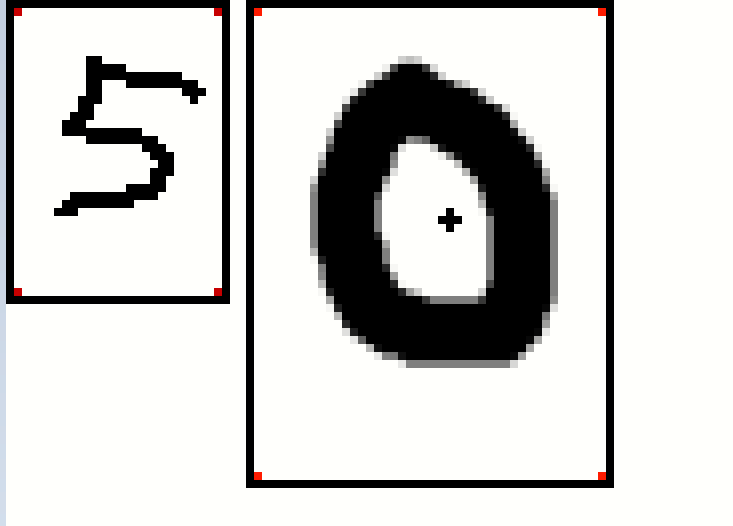


图 三-3 某一手写完成的样本

### 测试集构建

测试集样本的制作与训练集基本相同，不同的是，将所有图片都保存至同一文件夹“测试样本集”中，以供程序读取识别。

本次实验测试样本集如下图。

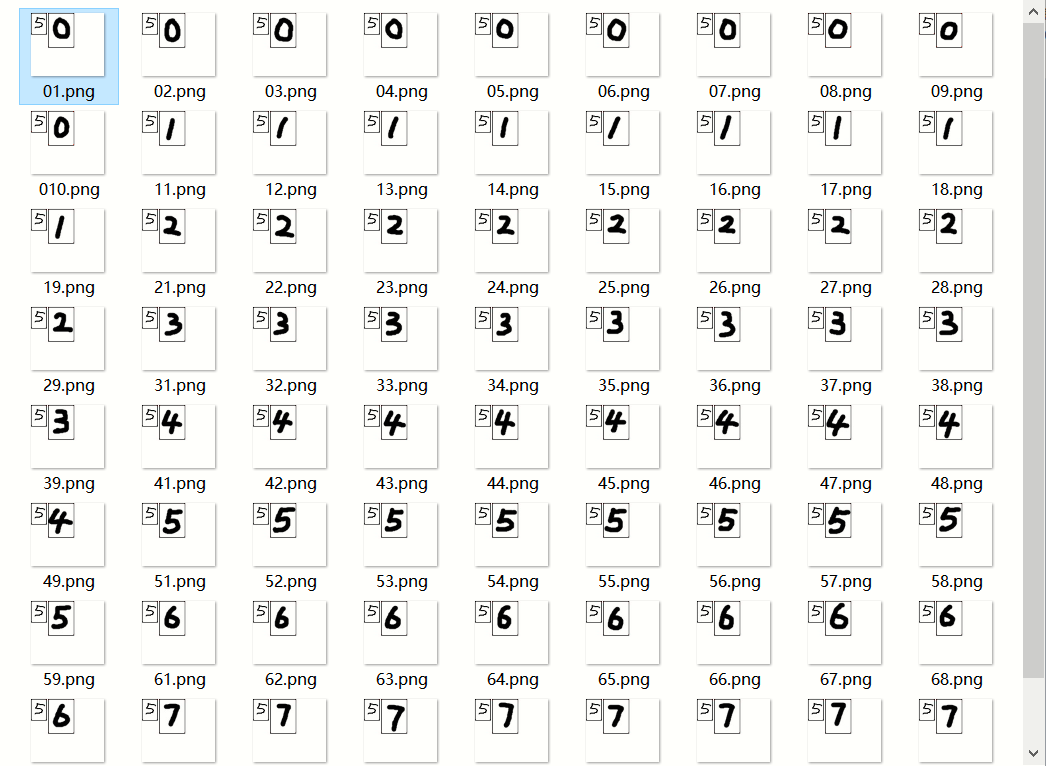


图 三-4 测试样本集

该样本集中有手写数字0-9各10个，它们的命名以各自的数字开头，用于实验中计算正确率。

### 实验结果

实验的运行环境是MATLAB GUI，本次实验各控件布局如下图。

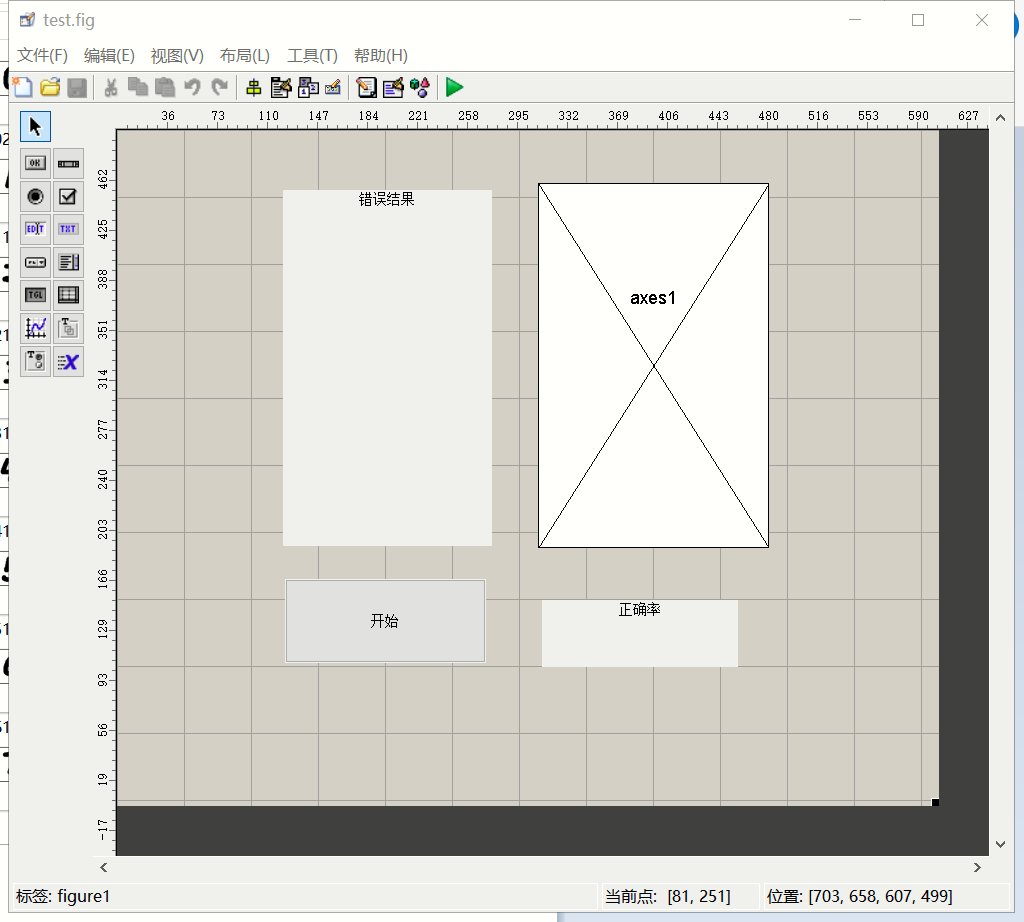


图 三-5 GUI控件的布局

图中已表明了各控件的作用，其中axes1用于运行时显示正在识别的样本。

程序运行效果如下：



图 6 程序正在运行

程序运行结果如下：

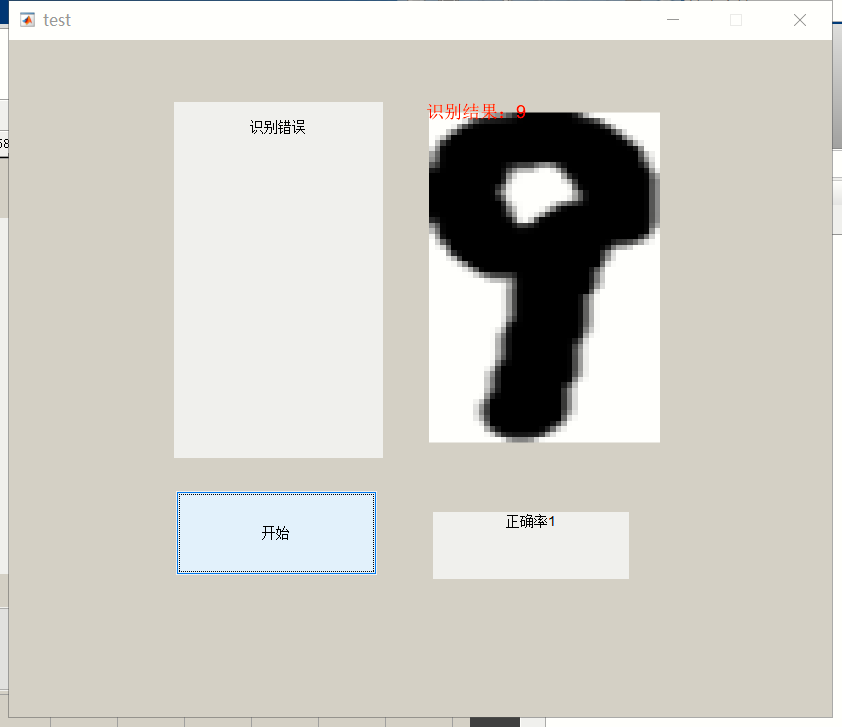


图 7 程序运行结束

### 性能量化评估

由程序运行结果可知，本次实验中对100个测试样本集的识别正确率达到了100%，验证该算法的有效性。

但，在实际的运行中发现，识别结果在每次训练中都不尽相同，即识别的结果具有不确定性，最严重时甚至可能出现代价函数无法稳定的情况，这些都与程序中大量使用随机函数有关。

## 结果讨论

#### 训练结果

本次实验随机选取了1000个样本对网络进行训练，在训练结束后，将代价函数与训练次数的关系可视化地表示出来，如下图：

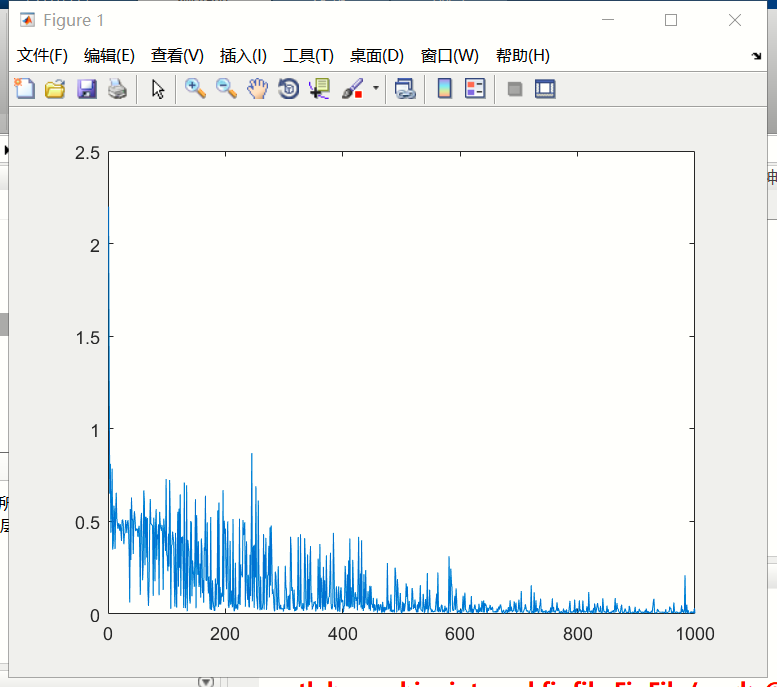


图 8 训练次数与代价函数的关系

从图中可以看到，随着训练次数的增加，代价函数基本上符合越来越小的设计目标，从而证明了该算法的有效性。可以推测，若进一步增加训练次数，代价函数将继续趋向于0并且“毛刺”会越来越少。

#### 识别结果

在实际测试过程中发现，对于特定的测试样本集，使用不同的训练结果可能会有不同的结果，识别的正确率从0.9~1不等，但增加训练次数可以增加准确率。

## 结论

本次实验构建了一个单隐层的神经网络，并且通过100个训练样本以及BP算法对该网络进行了训练，最后使用训练出的网络对测试样本集中的样本进行的测试。

虽然多次运行结果显示最终的实验结果不唯一，但识别的正确率基本在0.9~1之间，正确率较高，说明了BP算法的有效性。